文献阅读报告

### 论文信息

论文题目：**Matrix Embedding for Large Payloads**

作者：

* ***Jessica Fridrich***：宾汉顿大学电气与计算机工程系
* ***David Soukal***：宾汉顿大学计算机科学系

发表时间：**2006.9**

发表期刊：**IEEE Transactions on Information Forensics and Security**

### 论文内容摘要

矩阵编码是一种通用的编码方法，它能被用于大多数的隐写方案以提高其**[嵌入效率](#EmbeddingEfficiency)**。由于对载体图像更少的改变更不容易破坏其统计特性，因此采用矩阵编码的方案通常具有较好的隐写安全性。和短消息相比，这种增益对长消息来说更为重要，因为长消息被更容易被检测。之前的工作主要是利用汉明码的方式来进行矩阵编码，然而这种编码方式对于长消息来说效率不高。在这篇工作中，我们提出了新的矩阵编码方案，该方案在嵌入接近[**嵌入容量**](#EmbeddingEfficiency)的消息时仍然具有较高的嵌入效率。其中一种方案基于[**单纯形码**](#SimplexCode)，另一种方案基于小维度的随机线性码。

### 论文背景介绍

#### 应用背景

在**统计上不被检测**出来是隐写方案的主要需求。论文中提到了四种主要影响隐写安全性的因素：

1. 载体类型
2. 载体中可能被修改位置的选择方法
3. 嵌入操作
4. 嵌入效率

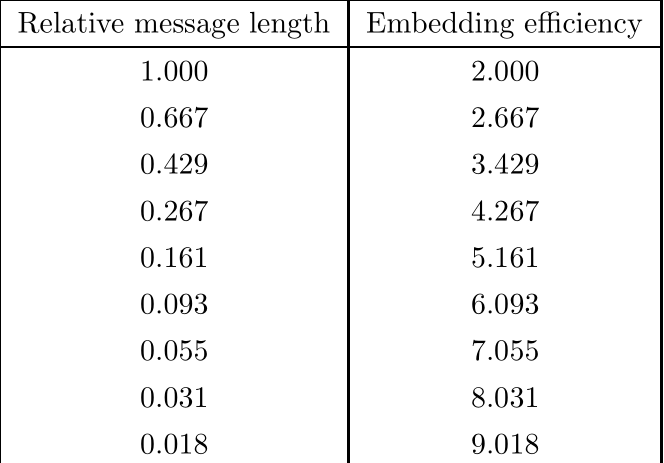
当前三种因素一样时，嵌入效率更高的方案被检测出来的可能性更低，因为对原图的修改更少。

**矩阵编码**能够有效的提高嵌入效率，从而提高载体在统计分析下的安全性。

#### 技术现状

矩阵编码最早由***Grandall***在1998年提出，接着***Bierbrauer***进行了分析。并且被***Willems***和***Galand***等人再次独立发现。***Westfeld***最先将矩阵编码应用到隐写算法**F5**中(在[拓展阅读](#_F5)部分会有详细介绍)。

**F5**算法结合了汉明码和**F4**算法，该算法当需要嵌入的消息较多时(大于嵌入容量的)，其对嵌入效率的改善较小，远达不到嵌入效率在理论上的上界。下图是其[**相对消息长度**](#RelativeMessageLength)和嵌入效率的变化关系：



可以发现随着相对消息长度的增加，嵌入效率逐渐减少，当相对消息长度大于时，嵌入效率降低为2。

#### 主要工作

本文针对基于汉明码的矩阵编码在**相对消息长度较大时嵌入效率较低**的缺点，提出了两种对于长消息更高效的矩阵嵌入方式：

1. 小维度生成矩阵随机的码。
2. 单纯形码

### 论文工作介绍

论文的主要工作在于提出了**随机线性码(random linear codes)**以及**单纯形码(simplex codes)**提高**嵌入率较高时**的嵌入效率。

#### 随机线性码

随机线性码基于码。

##### 编码流程

**嵌入**：

1. 使用伪随机算法从载体中随机读取bits形成的向量，读取下一个bits消息片段。
2. 计算。
3. 找到向量满足。
4. 穷举所有个**[码字](#Codeward)**，找到距离最近的码字。
5. 修改为
6. 如果没有更多的信息需要嵌入，停止，否则回到1。

**提取**：

1. 隐藏的信息。

##### 分析

在上述的步骤中，**最关键**也**最耗时**的步骤是嵌入算法的第四步。这一步保证了**对载体的改变最小**并且保证了提取的成功。其依赖于如下的结论：

***给定[陪集](#Coset)，对于任意的，有。更进一步的，如果存在，那么是的***[***陪集首***](#CosetLeader)***。***

根据上述的结论，我们有，其中是的**[奇偶校验矩阵](#ParityCheckMatrix)**。这保证了对载体的修改次数最少。因为奇偶校验矩阵是的[**对偶码**](#DualCode)的[**生成矩阵**](#GenerateMatrix)，因此我们有，这保证了，即提取的步骤是正确的。

需要注意的地方是尽管奇偶校验矩阵是随机生成的，但具有特定的格式：，其中是单位矩阵。这保证了嵌入过程中第三步找的过程是容易的。

#### 单纯形码

单纯形码是码，它具有如下的特性：

* 它是汉明码的[**对偶码**](#ParityCheckMatrix)。因此它的生成矩阵即是[汉明码](#_基于汉明码的矩阵编码)的奇偶校验矩阵。
* 它是恒重码。即所有的非零码字都具有相同的[**汉明权重**](#HammingWeight)。
* 任意两个[**码字之间的距离**](#VectorDistance)是。

上述的特性使得对于任意的，存在迅速的**解码算法(fast Hadamard transform)**能够找到离最近的码字。这克服了**随机线性码耗时**的问题。

##### 编码流程

**嵌入**：

1. 使用伪随机算法从载体中随机读取bits形成的向量，读取下一个bits消息片段。
2. 计算。
3. 找到向量满足。
4. 使用fast Hadamard transform找到与最近的码字。
5. 修改为。
6. 如果没有更多的消息需要嵌入，则停止，否则返回1。

**提取**：

1. 隐藏的信息。

##### 分析

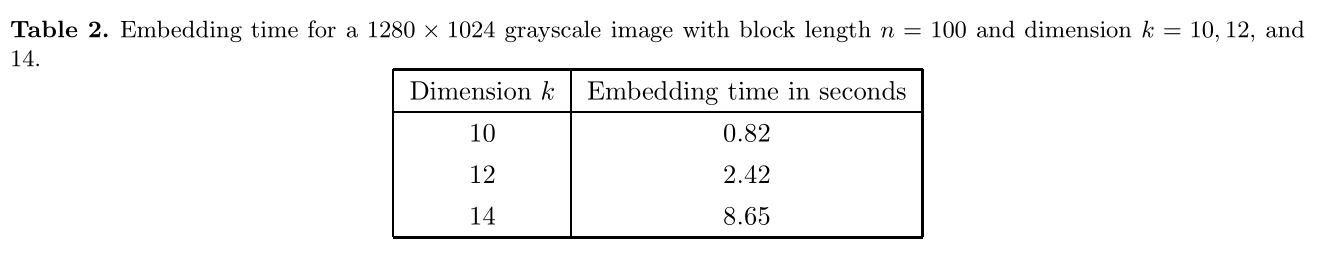
注意编码流程中与随机线性码仅有的不同是第4步，由于单纯形码的特殊性，使得其能够在的条件下迅速找出最近的码字，这解决了随机线性码中寻找码字速度慢的问题。

##### 加强

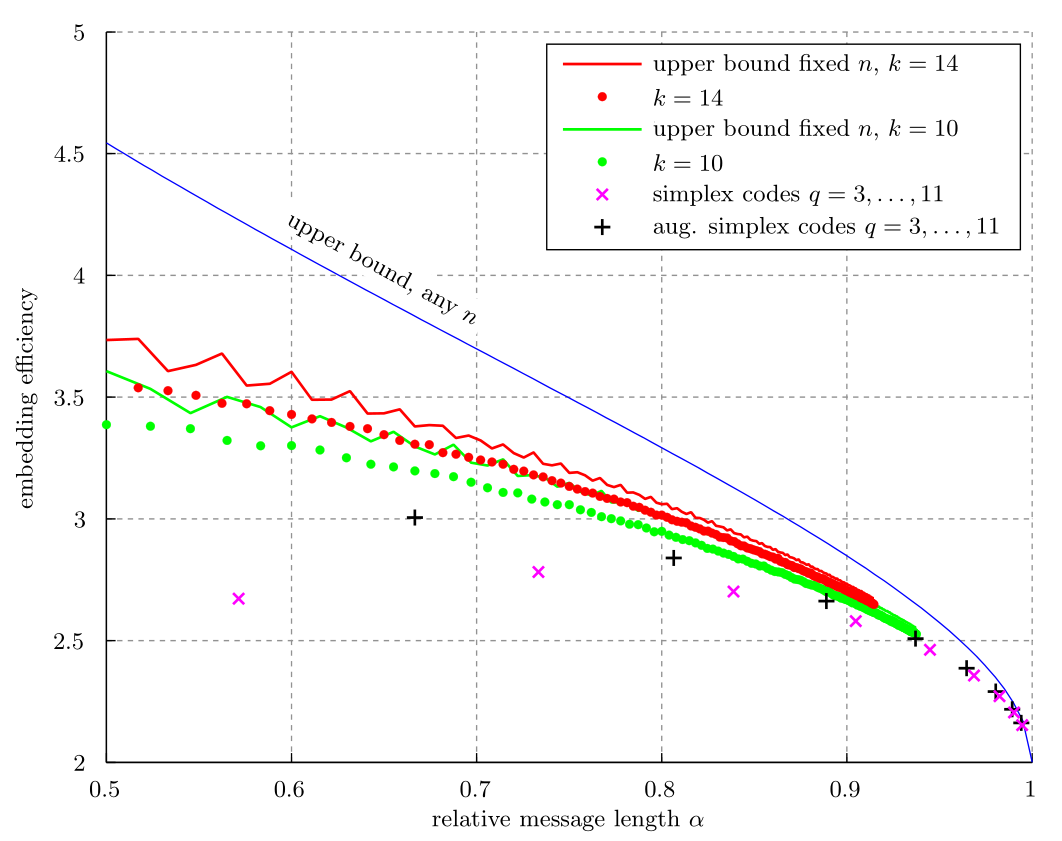
还有许多其他由单纯形码导出的码能够被用在矩阵编码中。他们在原始编码算法的基础上只需要少许的修改就能够在一定程度上提高嵌入效率。尽管如此，就嵌入效率而言，**随机线性码的嵌入效率最高**。单纯形码及其加强的形式可以看作**用嵌入效率来换时间**。

### 论文实验情况

下图是取不同值时，使用单纯形码进行消息嵌入的实验，可以发现随着维度的增加，需要的**嵌入时间呈指数形式增长**。



下图是在不同的编码方式下，相对消息长度(嵌入率)与嵌入效率的关系。其中**蓝色的线**表示任意时理论上嵌入效率的上界，**红线**是在时，码理论上嵌入效率的上界；**红点**是时使用随机线性码的嵌入效率，可以看到随机线性码**非常接近理论上码的上界**；**绿线**与**绿点**的含义类似于红线和红点；紫色乘号表示使用单纯形码的嵌入效率，黑色加号表示增强的单纯形码的嵌入效率。观察实验结果我们可以知道，随机线性码的嵌入效率最好，并且随着的增大，嵌入效率呈现增加的趋势，但代价是耗费的时间较多。单纯形码及其派生码嵌入效率略低于随机线性码，但编码花费的时间较少。在基于汉明码的矩阵编码中，随着相对消息长度的增加，其嵌入效率会迅速降低，当相对消息长度大于时，**嵌入效率与不使用矩阵编码相当**。而随机线性码和单纯形码较好的解决了该问题，当相对消息长度超过时，仍然具有较高的嵌入效率。



### 论文阅读总结

矩阵编码能够在不改变原始隐藏算法的情况下提高其嵌入效率。本文提出了两种矩阵编码的方式，这两种方式适用于**嵌入率较高**(嵌入的信息与嵌入容量接近)的情况。一种是随机线性码，该编码的特点是其嵌入效率能够逼近理论上的上界，但是较为耗时。另一种是基于单纯形码，该编码的嵌入效率稍弱于随机线性码，但相对花费较少的时间。

文章中提到的两种算法都改善了基于汉明码的矩阵编码在嵌入率较高情况下对嵌入效率提升有限的问题。两种算法可以看成时间和嵌入效率的**trade-off**，在某些对隐秘性要求较高的场合，我们可以使用随机线性码，用更多的时间确保更高的嵌入效率；而在某些对交互性要求较高的场合，我们可以使用单纯形码，确保交流的实时性。

### 学习笔记

**嵌入效率(embedding efficiency)**：对原始图像的每次改变能够嵌入的消息位数。例如LSB中修改嵌入消息的像素点的平均概率是，每个像素点能够嵌入bit的信息，即对原始图像的一次改变能够嵌入bit的信息，即该方案的嵌入效率是。**提高嵌入效率能够更小的破坏载体图像的统计特性，提高对统计分析攻击(statistical attack)的鲁棒性**。

**嵌入容量(embedding capacity)**：最高能够嵌入消息的比率。

**相对消息长度(relative message length)**：在某次嵌入的过程中，嵌入信息的比特数与该区域的pixel数量之比。例如某算法每次在个像素点上嵌入bits信息，那么其相对消息长度为。在***Westfeld***论文中，这个概念也被称之为**嵌入率(embedding rate)**。其实我感觉嵌入率可能更形象一些。

**单纯形码(simplex code)**：一组线性纠错码。

：所有的bit列向量形成的空间。如果的乘法和加法运算都是在有限域上时，称该控件为**线性向量空间(linear space vector)**。

**码字(codeword)**：的子集中的向量。

**汉明权重(hamming weight)**：向量中的个数，记作。

**向量之间的距离**：。

：以为中心，半径为的球，其含义为：

**向量与子集的距离**：，即到码字的最短距离。

**子集的覆盖范围**：，即到码字的最长距离。

**到的平均距离**：，即中所有的向量与距离的平均值，显然有。

**向量与子集的加法**：。

**子集的冗余度**：，是集合的势，即码字的个数。

**子集的维度**：这里的维度有点类似于线性代数中**秩**的概念，即码字中线性无关的向量的最大的个数。

**码**：长度为，维度为的子集。结合冗余度的概念，我们容易得知

**生成矩阵**：码中个线性无关的向量组成的矩阵。结合线性代数中秩的性质，我们很容易可以知道中码字即为**生成矩阵行向量的线性组合**。

**对偶码**：，是码，其生成矩阵是矩阵，且有，也成为的**奇偶校验矩阵(parity check matrix)**。

**syndrome of (这个不知道如何翻译)**：称为的**syndrome**。

**陪集**：，这里陪集的概念和抽象代数里面陪集的概念应该是相同的，因此我们有，由陪集的定义我们容易得出。

**陪集首**：陪集中具有最小汉明权重的向量，该向量可能不止一个。

### 拓展阅读

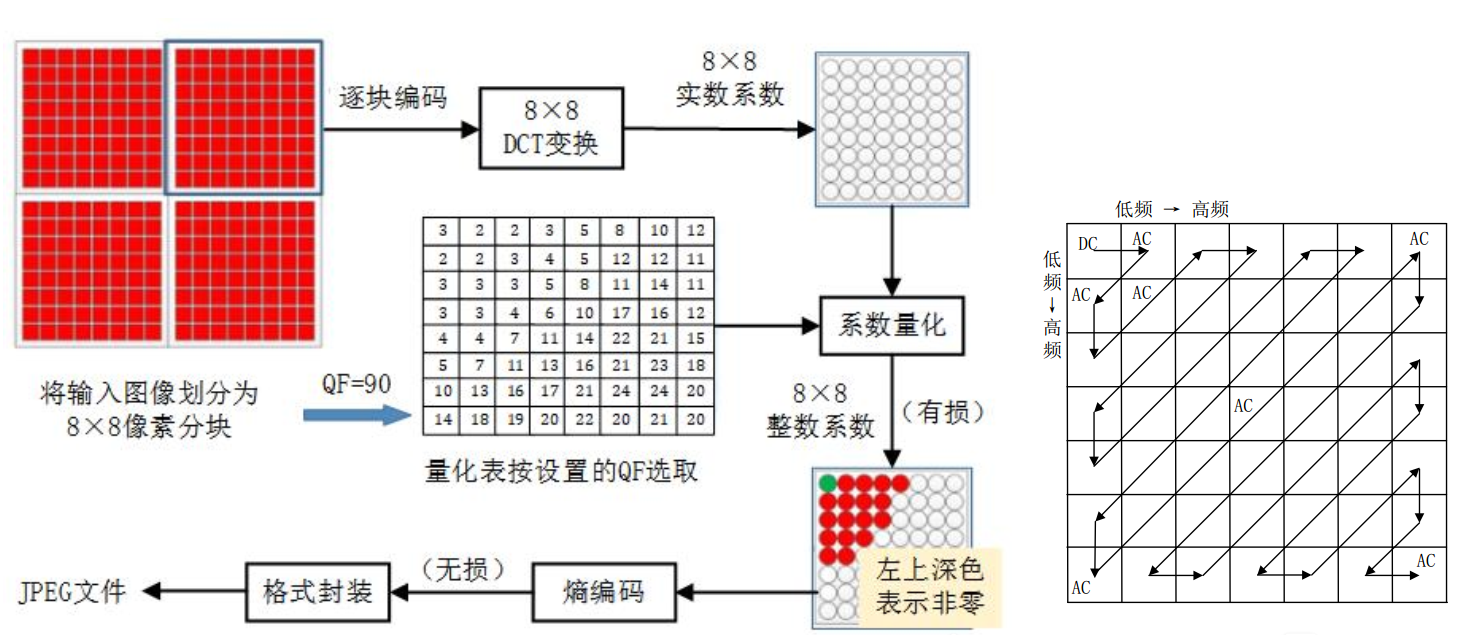
以下阅读主要基于文章

***High capacity despite better steganalysis (F5—A steganographic algorithm)***

在***Jessica***的文章中提到了**F5**隐写算法的对比，该算法由***Westfeld***提出，其在**F4**隐写算法的基础上增加了基于汉明码的矩阵编码用于提高效率。文章中还提到了一些其他的变换域隐写算法：包括**Jsteg**、**F3**以及**F4**。这些算法本质上都可以看作是在**Jsteg**上进行的优化。

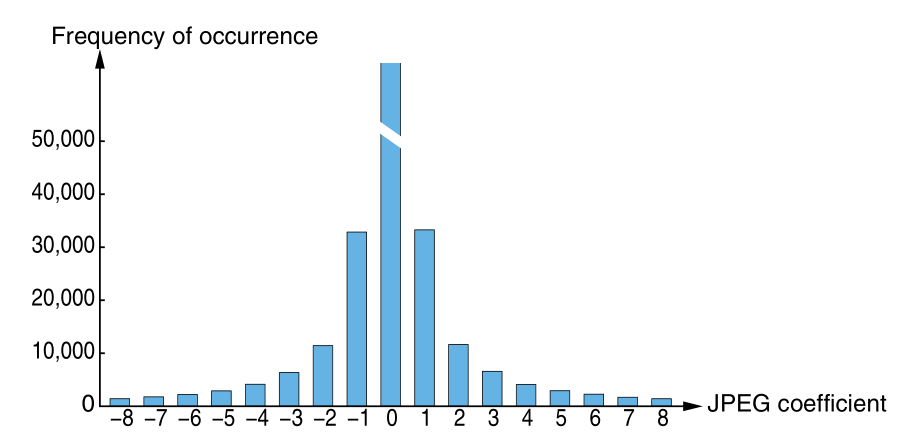
在分析这些算法之前，我们需要理解**JPEG**。

**JPEG**是一种针对照片影像而广泛使用的有损压缩标准方法，其主要步骤的流程图可以用下图来表示：



需要注意的是，在JPEG压缩的过程中，**量化(quantisation)**是一个有损的过程，它会把很多更高频率的成分舍位成为接近0，而上述提到的四种变换域压缩算法都是在**量化之后**进行的。

基于对变换域隐写算法的分析通常是统计分析，一种常用的方法是分析JPEG压缩过程中量化之后的系数直方图。对一张正常的图片来说，其量化后的系数直方图如下所示：



从直方图中我们能意识到三个重要的特性：

1. 系数0出现的频率最高，且随着系数绝对值的增加，出现的频率逐渐降低。
2. 随着系数绝对值的增加，出现的频率减少的越来越慢。这点可以理解为如果用一条曲线将直方图中的每个bar的中心点连接起来，则边缘部分会更加平滑，即下降的更缓慢。
3. 两侧分布近似对称。

#### Jsteg

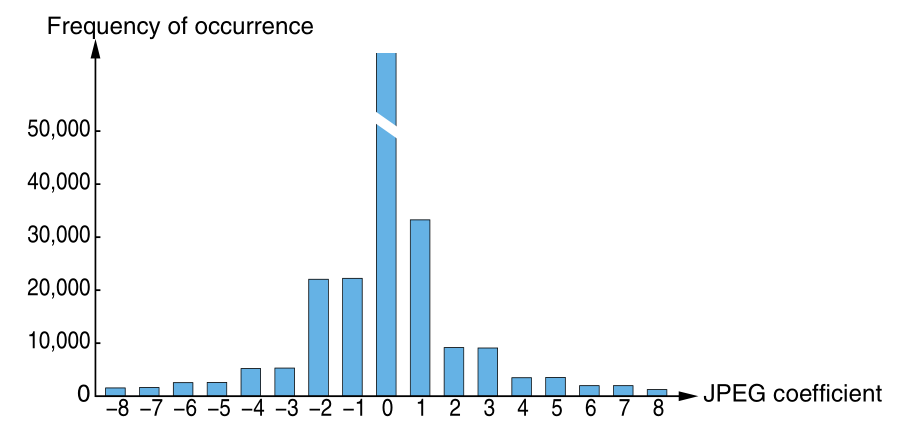
Jsteg算法的核心思想其实也是LSB，只不过和空域中直接对图像的像素点进行LSB隐写不同的是，Jsteg使用LSB的对象是量化后的DCT系数矩阵。其主要的算法流程如下：

1. 解码JPEG图像，得到量化后的DCT系数，判断该DCT系数**是否等于正负1或0**，若等于则跳过该DCT系数，否则，执行下一步。
2. 判断DCT系数的LSB是否与要嵌入的秘密信息比特相同，若相同，则不对其进行修改，否则执行下一步。
3. 用秘密信息比特替换DCT系数的LSB，将修改后的DCT系数重新编码得到隐秘JPEG图像。

上面的步骤流程中提到Jsteg不使用0以及正负1进行信息的隐藏，其原因主要有以下两点：

* DCT系数中0的比例最大（一般可达到60%以上，取决于图像质量和压缩因子），压缩编码是利用大量出现连零实现的，如果改变DCT系数中“0”的话，不能很好的实现压缩。
* DCT系数中的1若变成0，由于接受端**无法区分未使用的0和嵌入消息后得到的0**，从而无法实现秘密信息的提取。

Jsteg算法具有高嵌入容量()，并且能够抵御视觉攻击，但是由于其核心思想和LSB算法一致，因此也会产生**值对效应**的问题，对于统计分析(卡方、RS等)并不能很好的防御，下图是其修改后AC系数的频率直方图，我们可以较为明显的除0外，相邻的偶数和奇数之间其出现的频率非常接近。

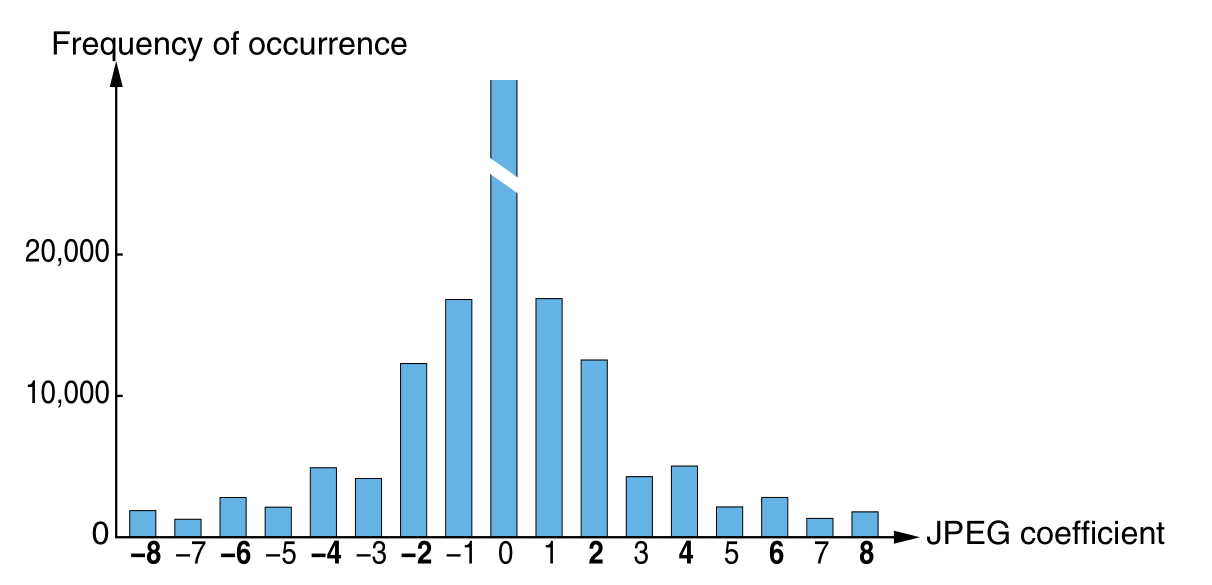


#### F3

根据DCT系数的频率直方图我们可以直观的观察到0以及正负1占据了系数中的一大部分，**而Jsteg中并没有利用这些系数**。F3算法即是为了解决该问题而出现的。该算法的主要步骤和Jsteg类似，只是隐写规则稍有区别：

1. 每个非零的DCT系数用于隐藏1bit信息，为0的系数不负载秘密信息。
2. 如果秘密信息与DCT的LSB相同，便不作改动；如果不同，**将DCT系数的绝对值减小1，符号不变**。
3. 当原始值为正负1且要嵌入秘密信息为0时，将这个位置**归0并视为无效**，在下一个DCT系数上重新嵌入。

F3的DCT系数直方图如下所示：



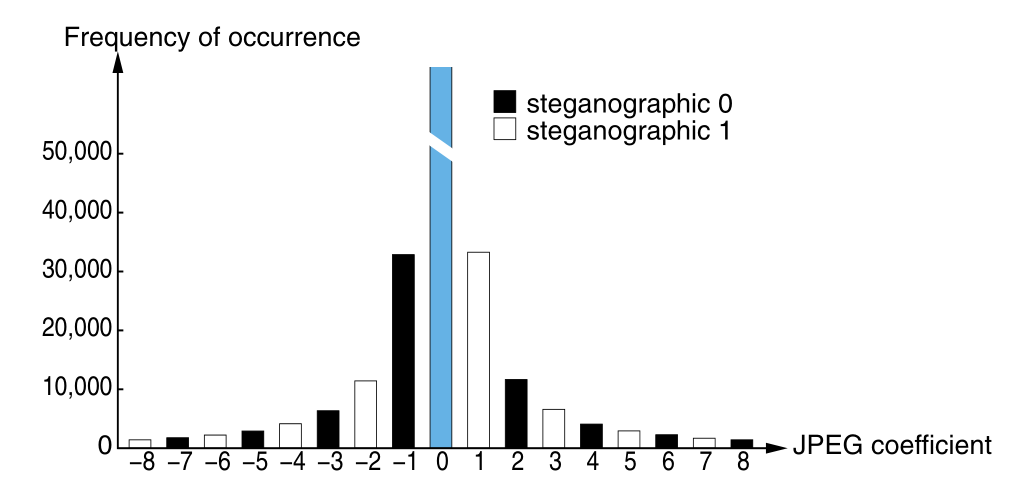
观察F3的直方图可以发现，其防止了**值对效应**，并且在一定程度上维持了分布函数的对称性。但使得偶数的分布增加，不满足原始直方图的第一个特性，即单调性；并且分布曲线呈现向零收缩的趋势。这使得F3对于统计分析仍然是有迹可循的。

#### F4

和F3类似，F4也通过减小绝对值的方法进行修改，但为了克服F3的系数直方图不满足单调性的缺点，F4对不同的正负号的奇偶系数采用了不同的嵌入与表示方法：

* 负偶数、正奇数代表嵌入消息1。
* 负奇数、正偶数代表嵌入消息0。

F4的DCT系数直方图如下所示：



可以发现F4的DCT系数直方图与原始图像DCT系数的直方图较为接近。

#### F5

在不使用矩阵编码的情况下，对于F4隐写算法而言，由于修改为0的位不能嵌入信息，其嵌入效率可能比2更小。***Westfeld***在F4算法的基础上结合了位置置乱和矩阵编码，较为明显的改善了载体的嵌入效率。论文中给出了一个短消息的例子，当嵌入bits信息时，使用F4隐写算法需要次修改，而使用F5隐写算法只需要次修改。

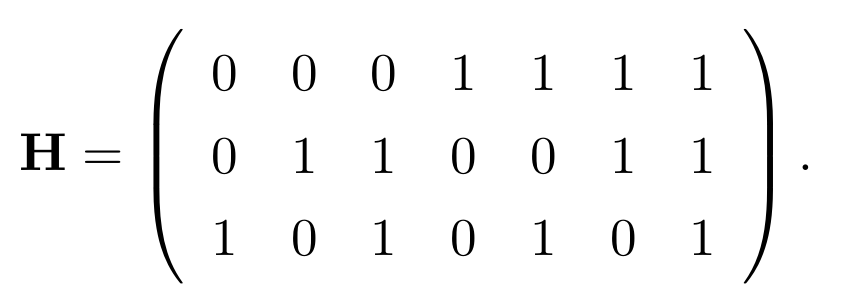
F5算法嵌入的主要步骤如下：

1. **打乱隐写位置(Permutative Straddling)**：该步骤的目的是让载体中的**嵌入浓度(embedding density)**尽可能相同，即载体中的嵌入位置尽可能的均匀。可以通过伪随机数发生器实现。
2. **矩阵编码(Matrix Embedding)**：该步骤的目的是提高**嵌入效率**。嵌入效率的提高能够使得在嵌入相同信息的情况下对载体的修改更小，防御统计分析的可能性更高。

##### 基于汉明码的矩阵编码

F5算法中使用的矩阵编码是基于**汉明码**，其之所以能够提高嵌入效率依赖于汉明码的如下特点：

1. **汉明码**是的线性码
2. 其**奇偶校验矩阵**具有如下性质：
   1. 矩阵的形状为
   2. 列向量是的**binary expansion**(二进制表示)。例如时的奇偶校验矩阵如下图所示：



假定载体图像一共有个pixel，我们首先将该图像划分为个子集，其中，那么对于每个子集，我们能够在不超过一次嵌入改变的情况下嵌入bits的信息。为了证明的方便，我们假定嵌入算法为LSB。

设个pixel的LSB组成的向量为；嵌入的bits消息组成的向量为。则**我们将用**来代替。其中代表的陪集首。当时，有，此时不需要进行修改，当时，，其中为的位是**将二进制向量转化为其表示的整数的位**。

由上述的结论我们可以知道如果使用LSB+基于汉明码的矩阵编码的方式，其嵌入效率为，嵌入率为。